FIG.1

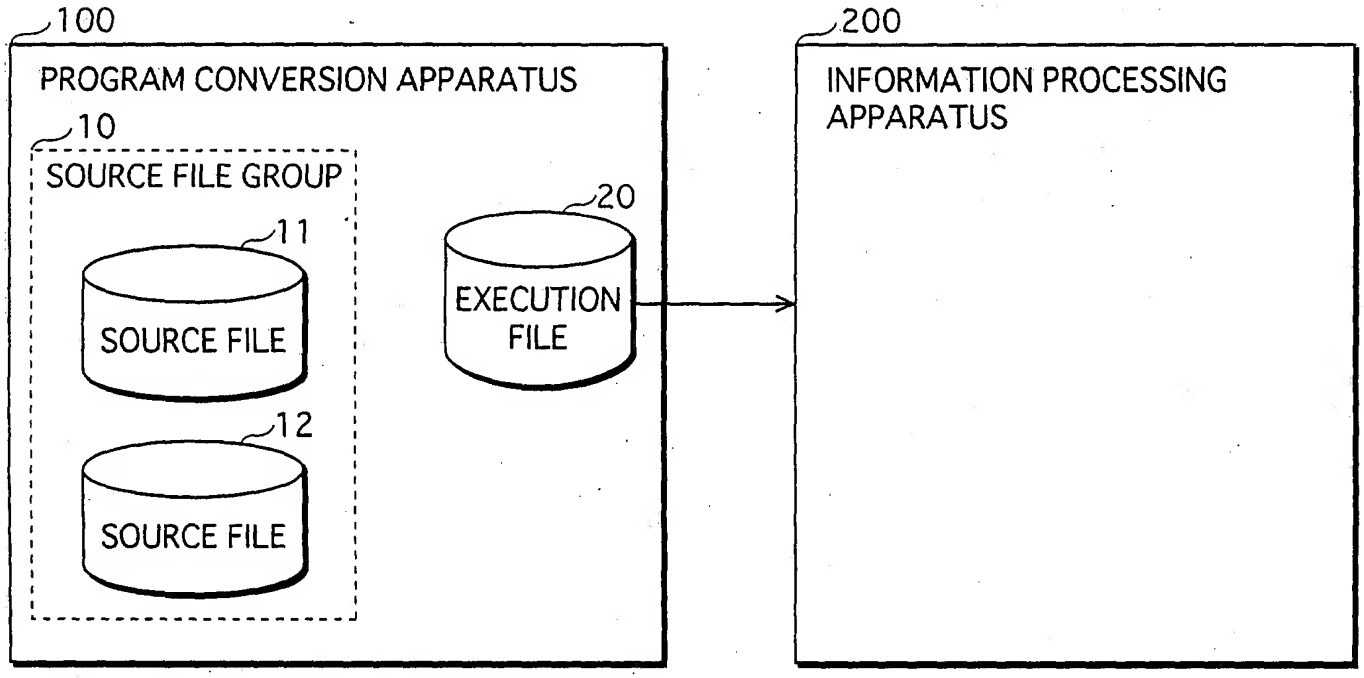

100
PROGRAM CONVERSION APPARATUS
10
SOURCE FILE GROUP
11
SOURCE FILE
12
SOURCE FILE
20
EXECUTION FILE
200
INFORMATION PROCESSING APPARATUS

FIG.2

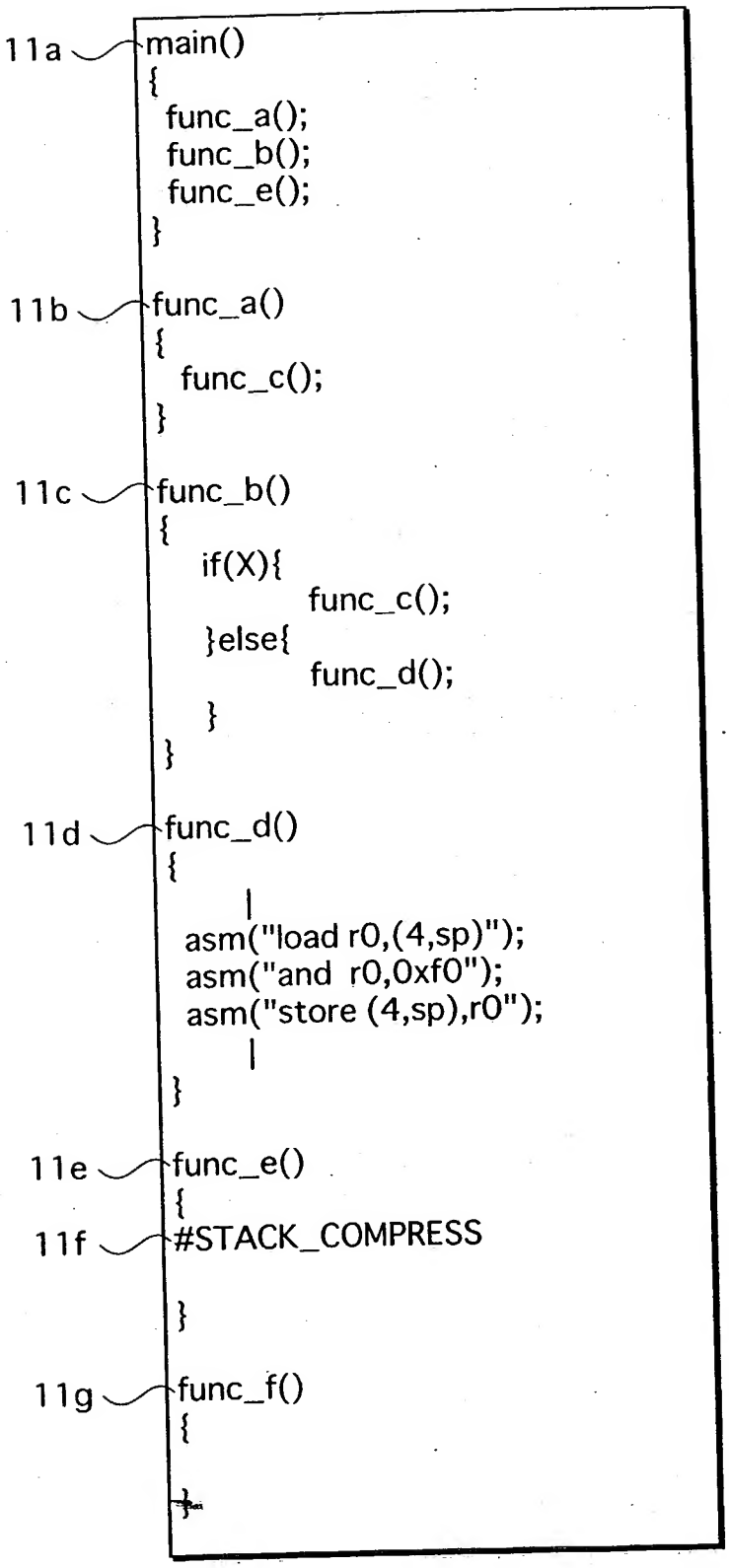

11a
main()
{
func_a();
func_b();
func_e();
}
11b
func_a()
{
func_c();
}
11c
func_b()
{
if(X){
func_c();
}else{
func_d();
}
}
11d
func_d()
{
asm("load r0,(4,sp)");
asm("and r0,0xf0");
asm("store (4,sp),r0");
}
11e
func_e()
{
11f
#STACK_COMPRESS
}
11g
func_f()
{
}

FIG.3

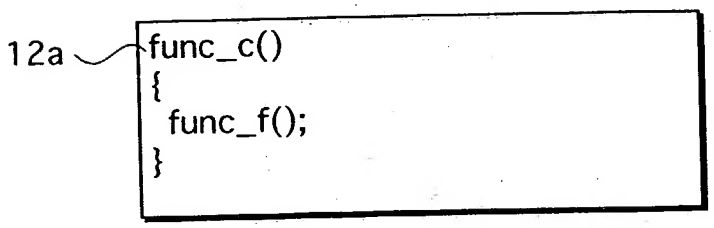


FIG.4

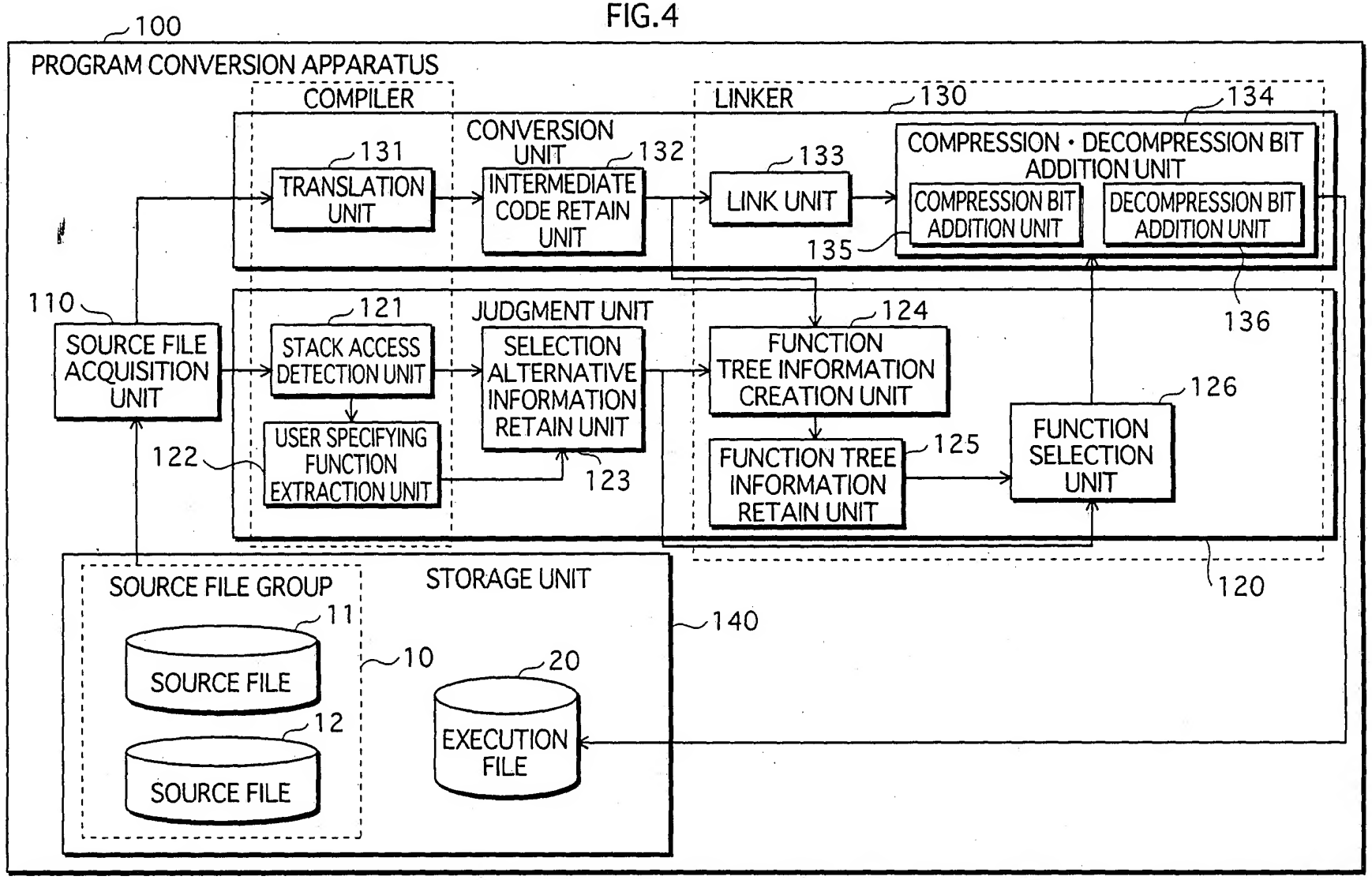

100
PROGRAM CONVERSION APPARATUS
COMPILER
LINKER
130
134
CONVERSION UNIT
131
TRANSLATION UNIT
132
INTERMEDIATE CODE RETAIN UNIT
133
LINK UNIT
COMPRESSION・DECOMPRESSION BIT ADDITION UNIT
COMPRESSION BIT ADDITION UNIT
DECOMPRESSION BIT ADDITION UNIT
135
136
110
SOURCE FILE ACQUISITION UNIT
121
STACK ACCESS DETECTION UNIT
JUDGMENT UNIT
SELECTION ALTERNATIVE INFORMATION RETAIN UNIT
124
FUNCTION TREE INFORMATION CREATION UNIT
126
122
USER SPECIFYING FUNCTION EXTRACTION UNIT
123
FUNCTION TREE INFORMATION RETAIN UNIT
125
FUNCTION SELECTION UNIT
120
SOURCE FILE GROUP
STORAGE UNIT
140
11
SOURCE FILE
10
20
12
SOURCE FILE
EXECUTION FILE

FIG.5

123a 123b 123c

FUNCTION NAME	FUNCTION EVALUATION VALUE
func_a	1
func_b	1
func_c	1
func_d	0
func_e	2
func_f	1

FIG.6

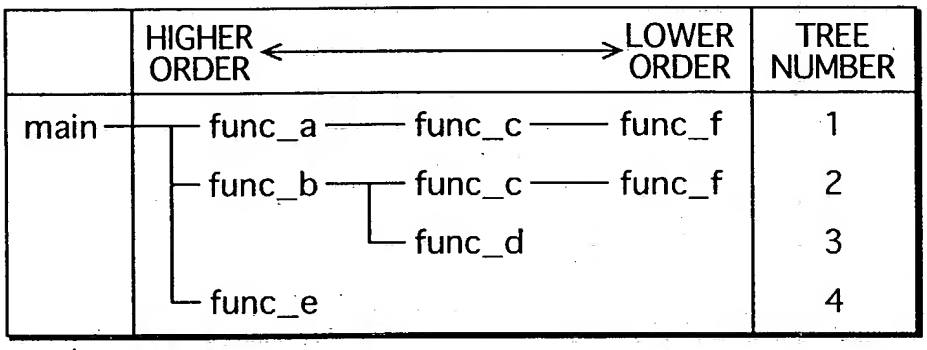

HIGHER ORDER
LOWER ORDER
TREE NUMBER
main
func_a
func_c
func_f
1
func_b
func_c
func_f
2
func_d
3
func_e
4

FIG.7

125a TREE NUMBER	125b TREE EVALUATION VALUE 125c
1	1 (=1×1×1)
2	1 (=1×1×1)
3	0 (=1×0)
4	2 (=2)

FIG.8

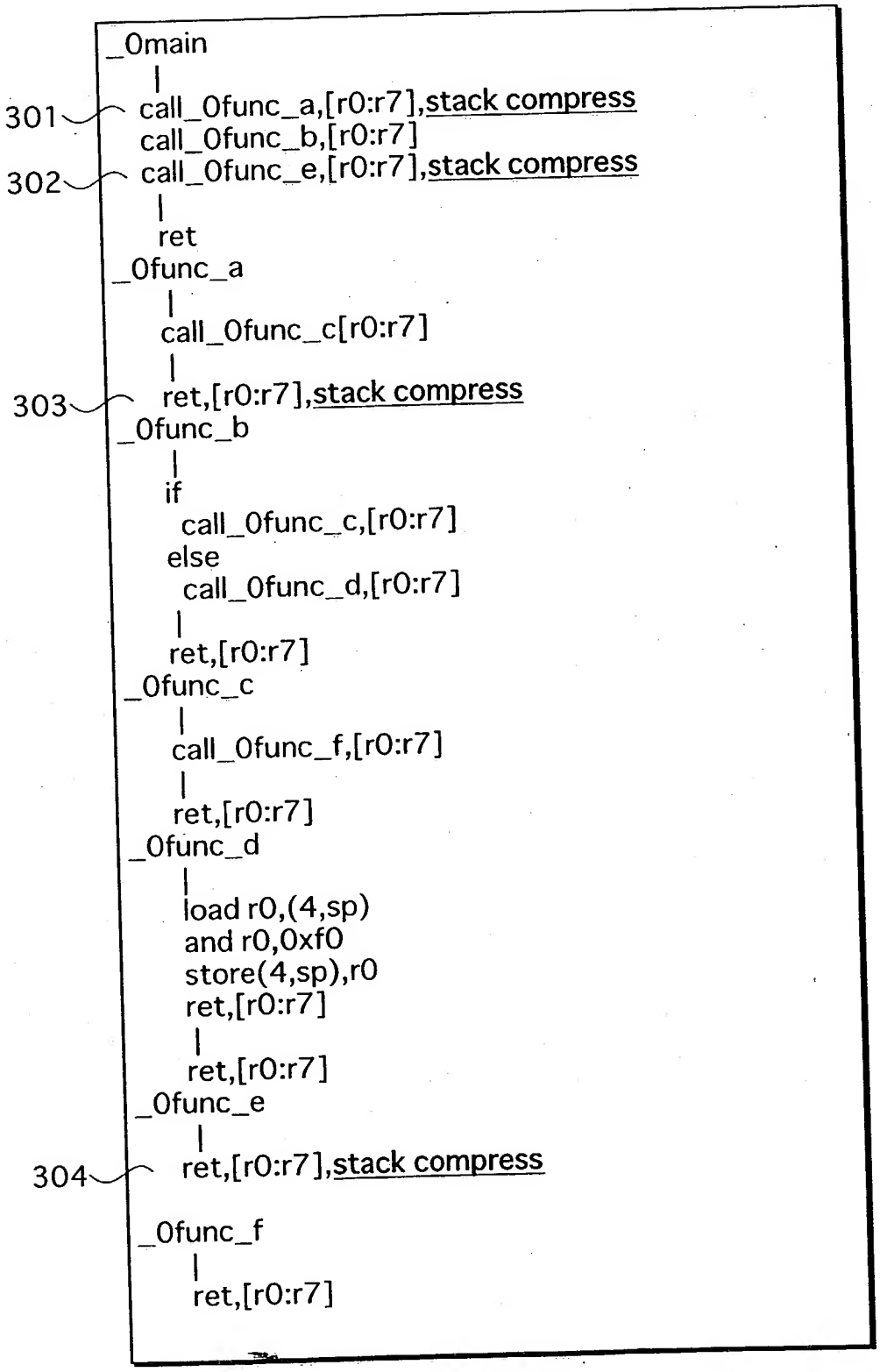

_0main
call_0func_a,[r0:r7],stack compress
301
call_0func_b,[r0:r7]
call_0func_e,[r0:r7],stack compress
302
ret
_0func_a
call_0func_c[r0:r7]
ret,[r0:r7],stack compress
303
_0func_b
if
call_0func_c,[r0:r7]
else
call_0func_d,[r0:r7]
ret,[r0:r7]
_0func_c
call_0func_f,[r0:r7]
ret,[r0:r7]
_0func_d
load r0,(4,sp)
and r0,0xf0
store(4,sp),r0
ret,[r0:r7]
ret,[r0:r7]
_0func_e
ret,[r0:r7],stack compress
304
_0func_f
ret,[r0:r7]

FIG.9

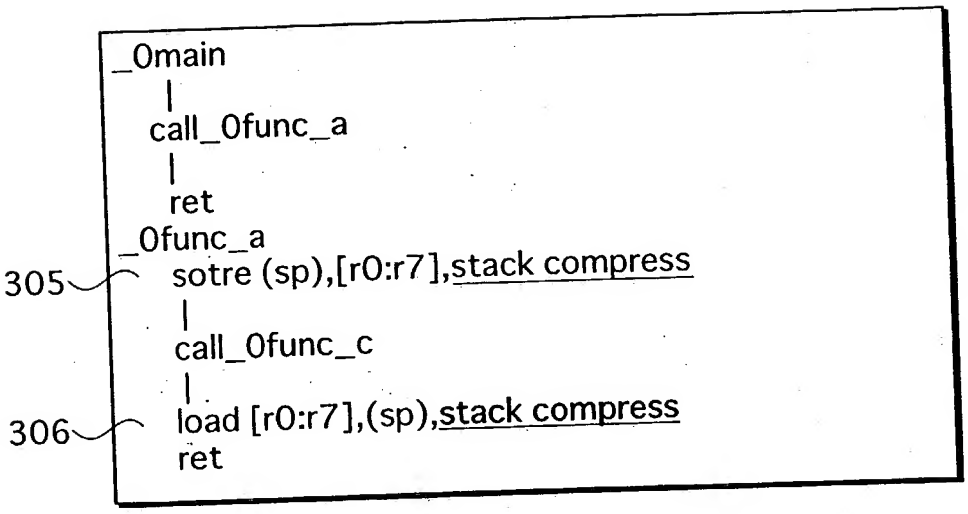

_0main
call_0func_a
ret
_0func_a
305
sotre (sp),[r0:r7],stack compress
call_0func_c
306
load [r0:r7],(sp),stack compress
ret

FIG.10

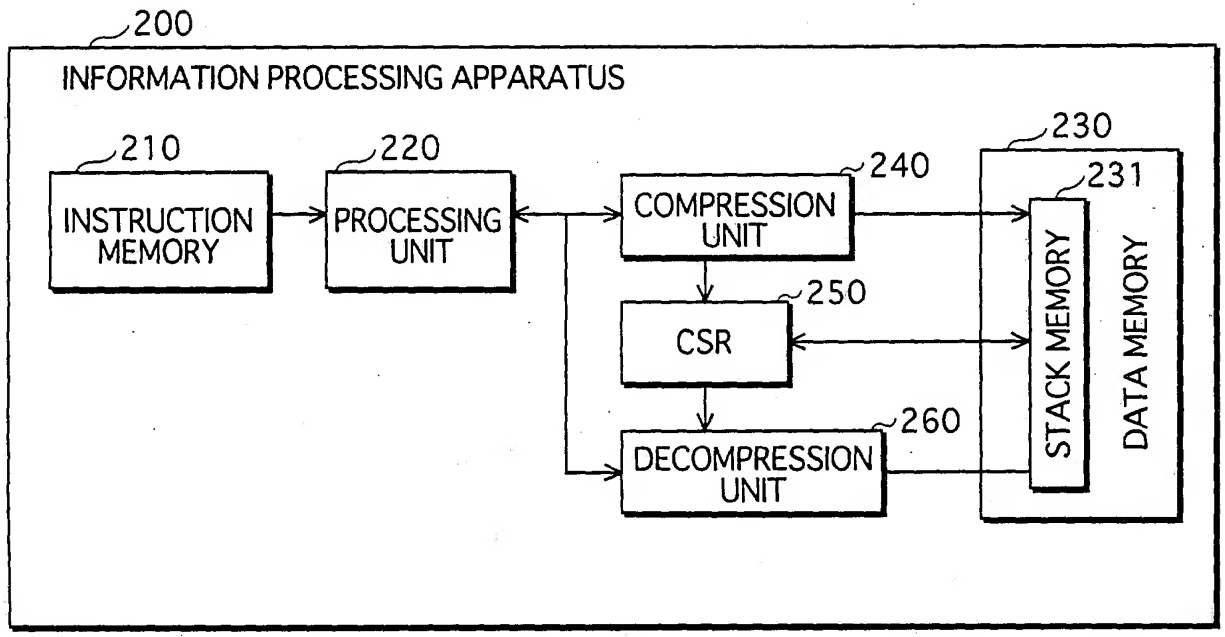

200
INFORMATION PROCESSING APPARATUS
210
INSTRUCTION MEMORY
220
PROCESSING UNIT
240
COMPRESSION UNIT
250
CSR
260
DECOMPRESSION UNIT
230
231
STACK MEMORY
DATA MEMORY

FIG.11

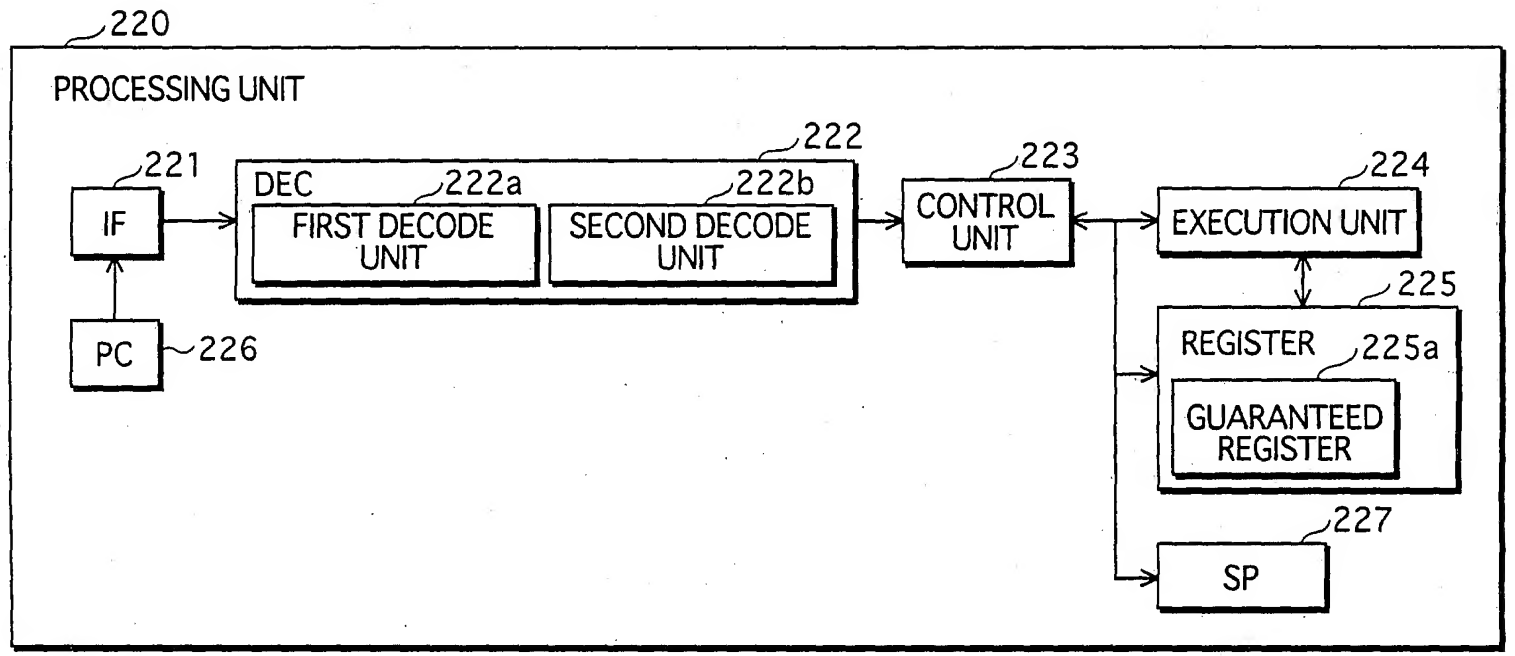

220
PROCESSING UNIT
221
IF
PC
226
222
DEC
222a
FIRST DECODE UNIT
222b
SECOND DECODE UNIT
223
CONTROL UNIT
224
EXECUTION UNIT
225
REGISTER
225a
GUARANTEED REGISTER
227
SP

FIG.12

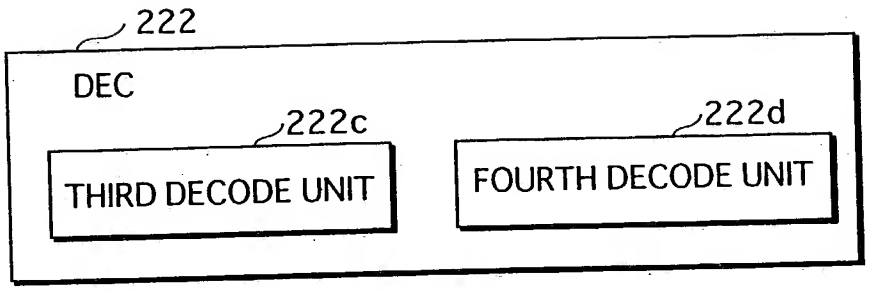


FIG.13

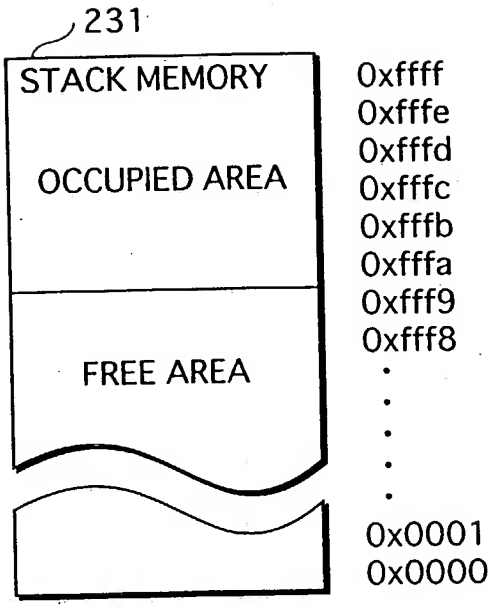

231
STACK MEMORY
OCCUPIED AREA
FREE AREA
0xffff
0xfffe
0xfffd
0xfffc
0xfffb
0xfffa
0xfff9
0xfff8
0x0001
0x0000

FIG.14

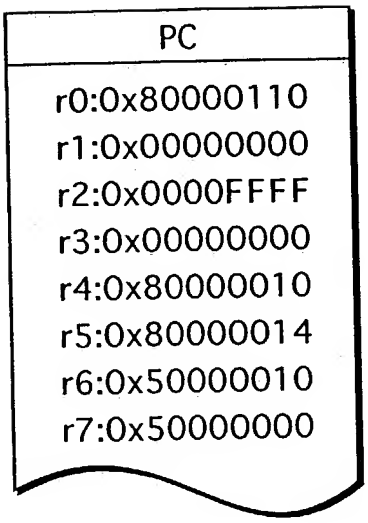

PC
r0:0x80000110
r1:0x00000000
r2:0x0000FFFF
r3:0x00000000
r4:0x80000010
r5:0x80000014
r6:0x50000010
r7:0x50000000

FIG.15

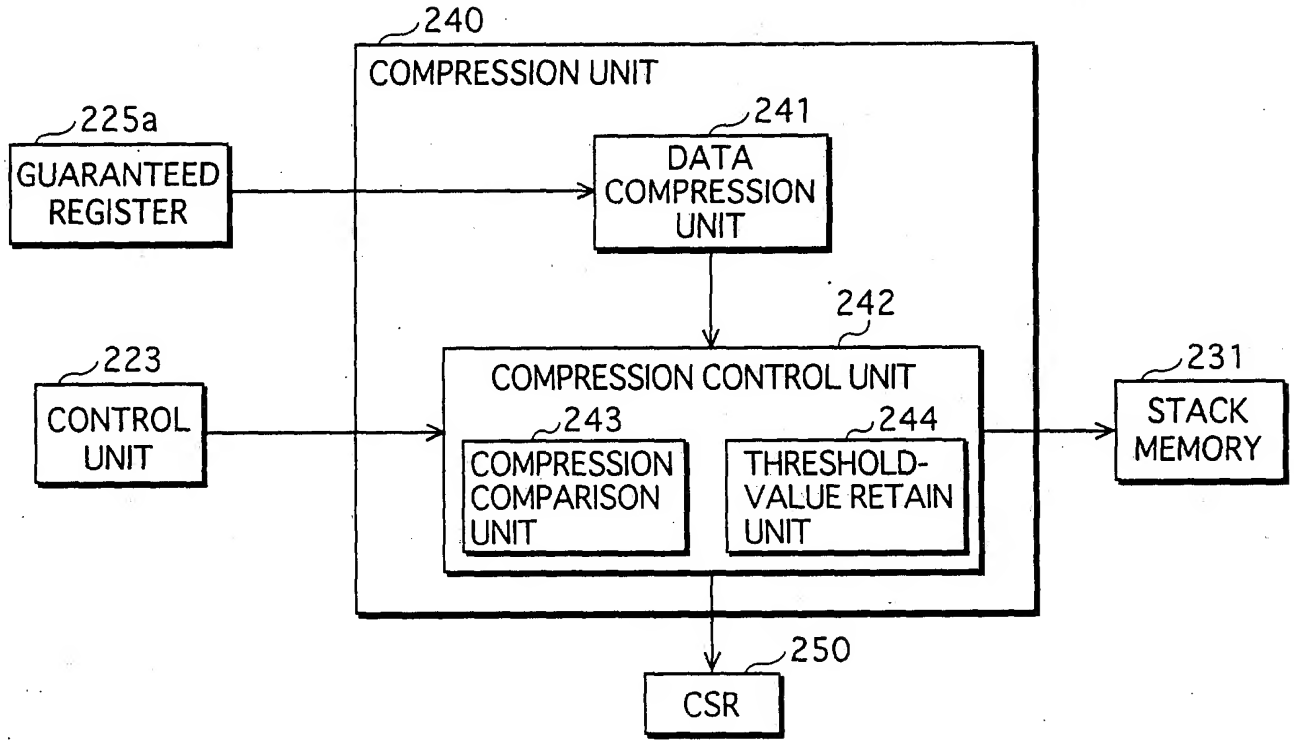

240
COMPRESSION UNIT
225a
GUARANTEED REGISTER
241
DATA COMPRESSION UNIT
242
223
CONTROL UNIT
COMPRESSION CONTROL UNIT
243
COMPRESSION COMPARISON UNIT
244
THRESHOLD-VALUE RETAIN UNIT
231
STACK MEMORY
250
CSR

FIG.16

VALUE	HUFFMAN-CODE
0x4	0b00000
0x5	0b00001
0x8	0b0001
0xf	0b001
0x1	0b01
0x0	0b1

FIG.17

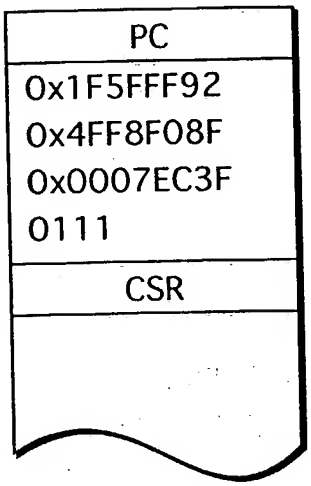


FIG.18

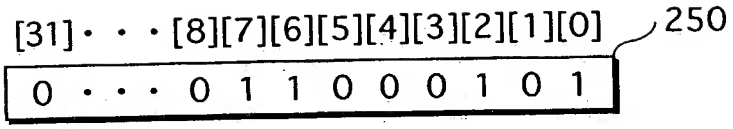


FIG.19

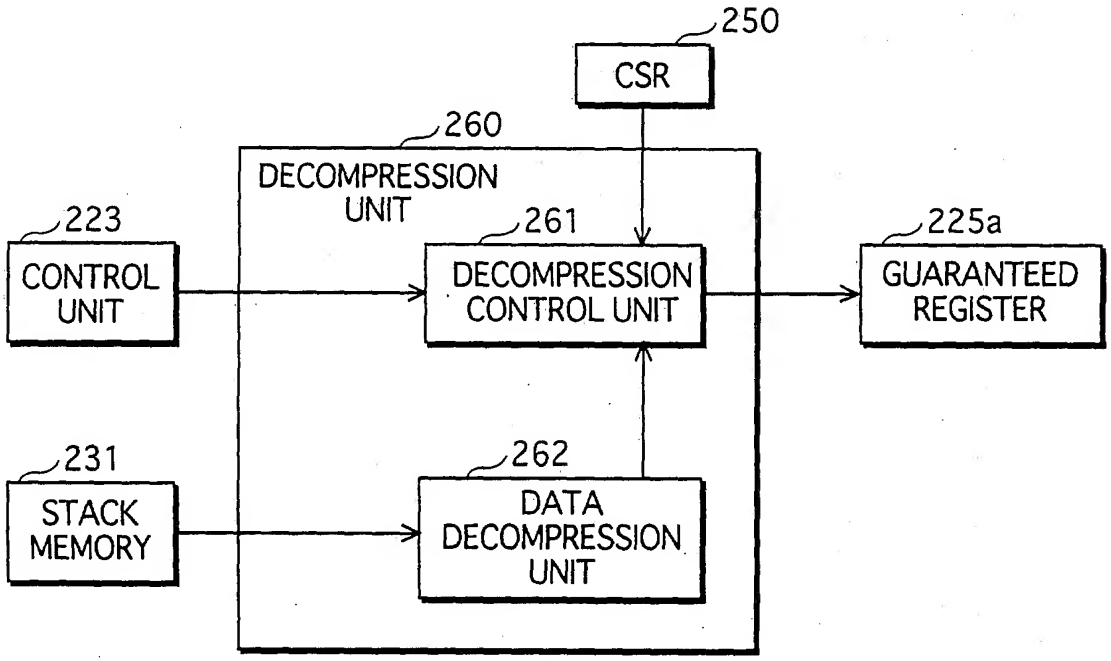

250
CSR
260
DECOMPRESSION UNIT
223
CONTROL UNIT
261
DECOMPRESSION CONTROL UNIT
225a
GUARANTEED REGISTER
231
STACK MEMORY
262
DATA DECOMPRESSION UNIT

FIG.20

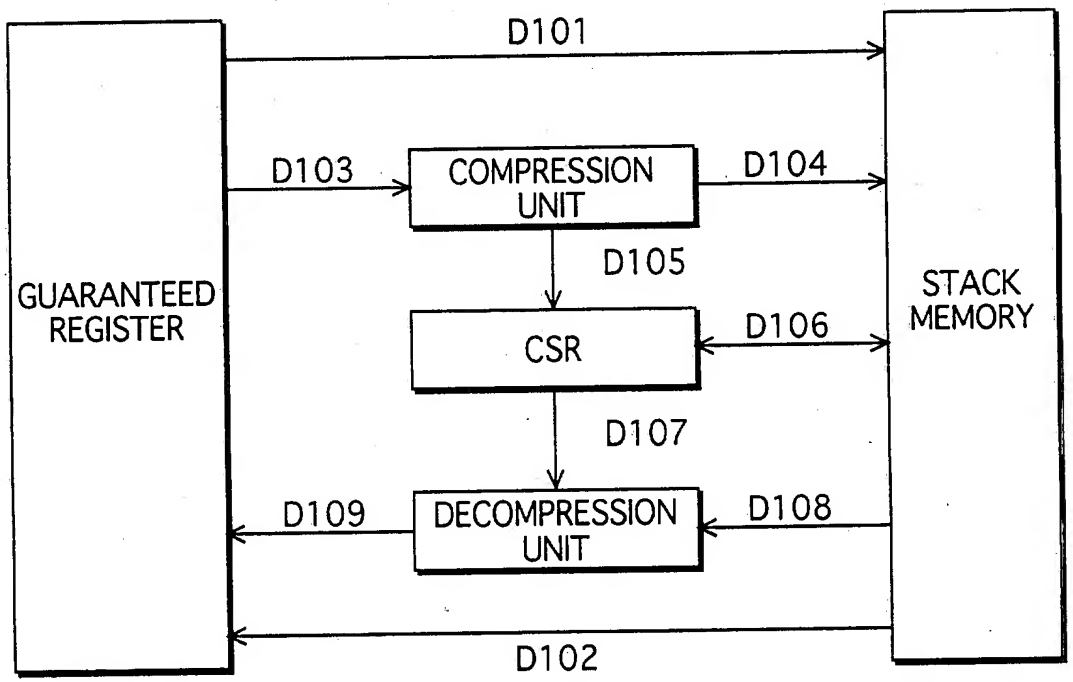

D101
GUARANTEED REGISTER
D103
COMPRESSION UNIT
D104
D105
CSR
D106
STACK MEMORY
D107
D109
DECOMPRESSION UNIT
D108
D102

FIG.21

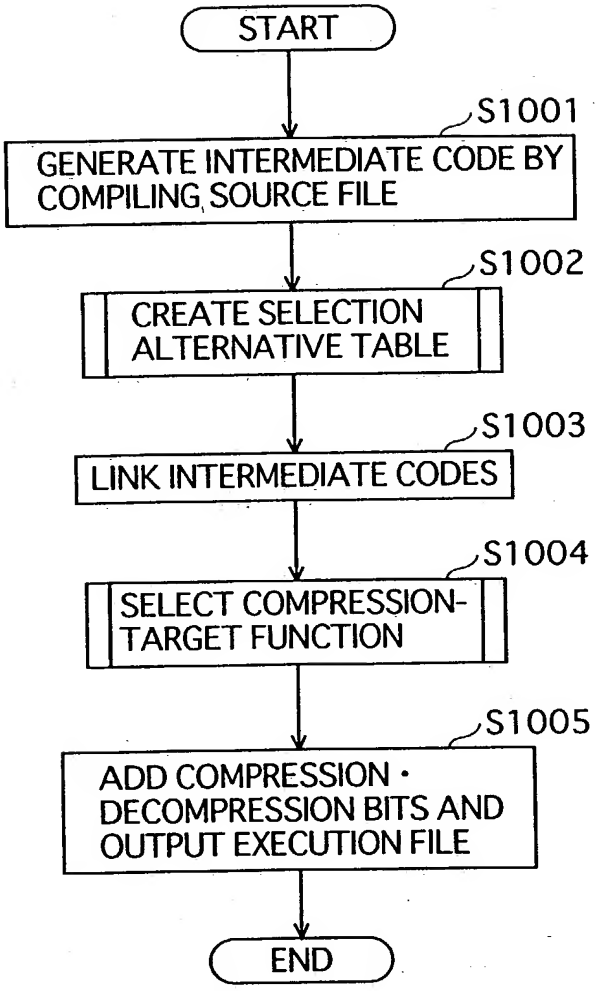

START
S1001
GENERATE INTERMEDIATE CODE BY COMPILING SOURCE FILE
S1002
CREATE SELECTION ALTERNATIVE TABLE
S1003
LINK INTERMEDIATE CODES
S1004
SELECT COMPRESSION-TARGET FUNCTION
S1005
ADD COMPRESSION・DECOMPRESSION BITS AND OUTPUT EXECUTION FILE
END

FIG.22

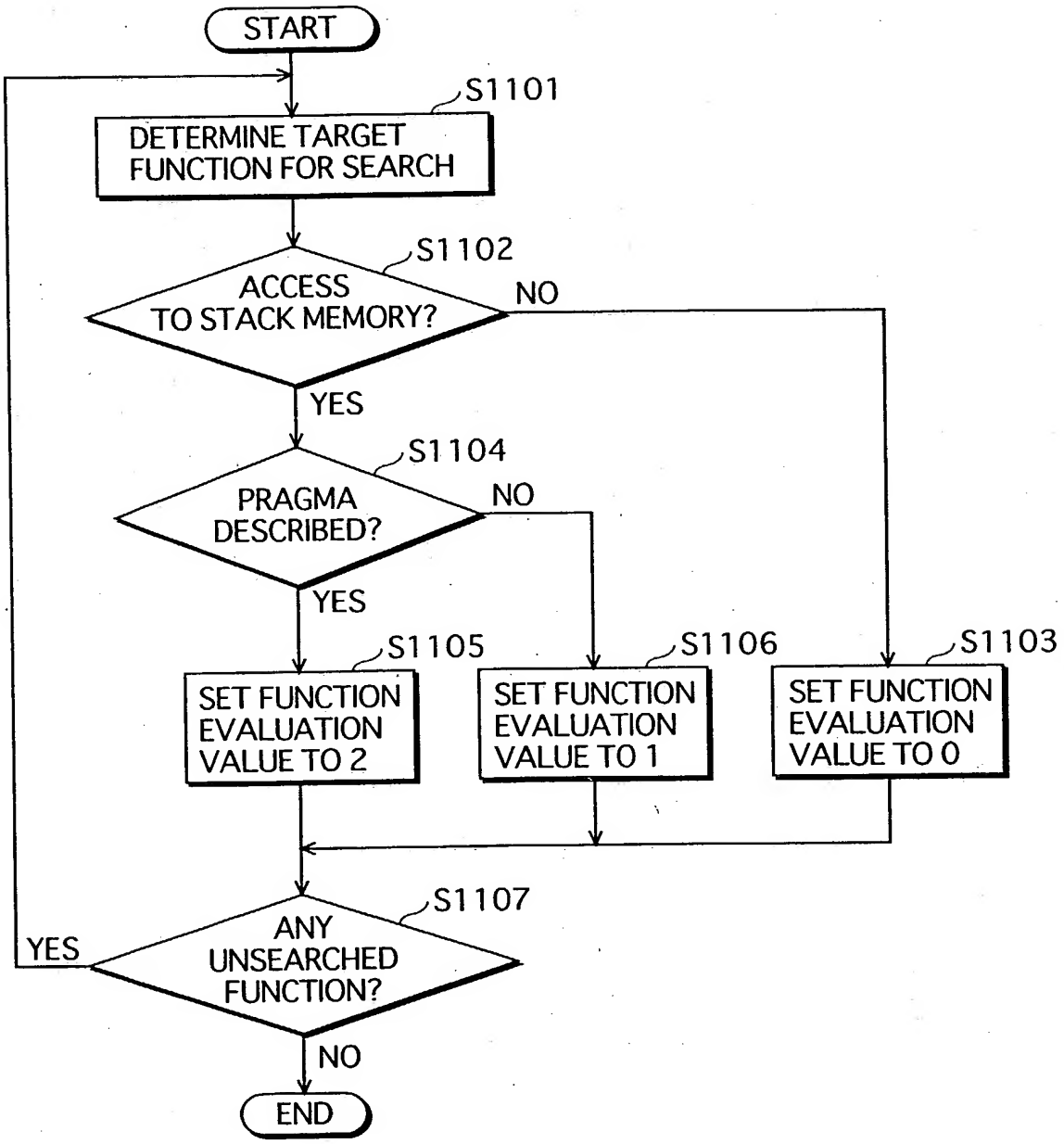

START
S1101
DETERMINE TARGET FUNCTION FOR SEARCH
S1102
ACCESS TO STACK MEMORY?
NO
YES
S1104
PRAGMA DESCRIBED?
NO
YES
S1105
SET FUNCTION EVALUATION VALUE TO 2
S1106
SET FUNCTION EVALUATION VALUE TO 1
S1103
SET FUNCTION EVALUATION VALUE TO 0
S1107
ANY UNSEARCHED FUNCTION?
YES
NO
END

FIG.23

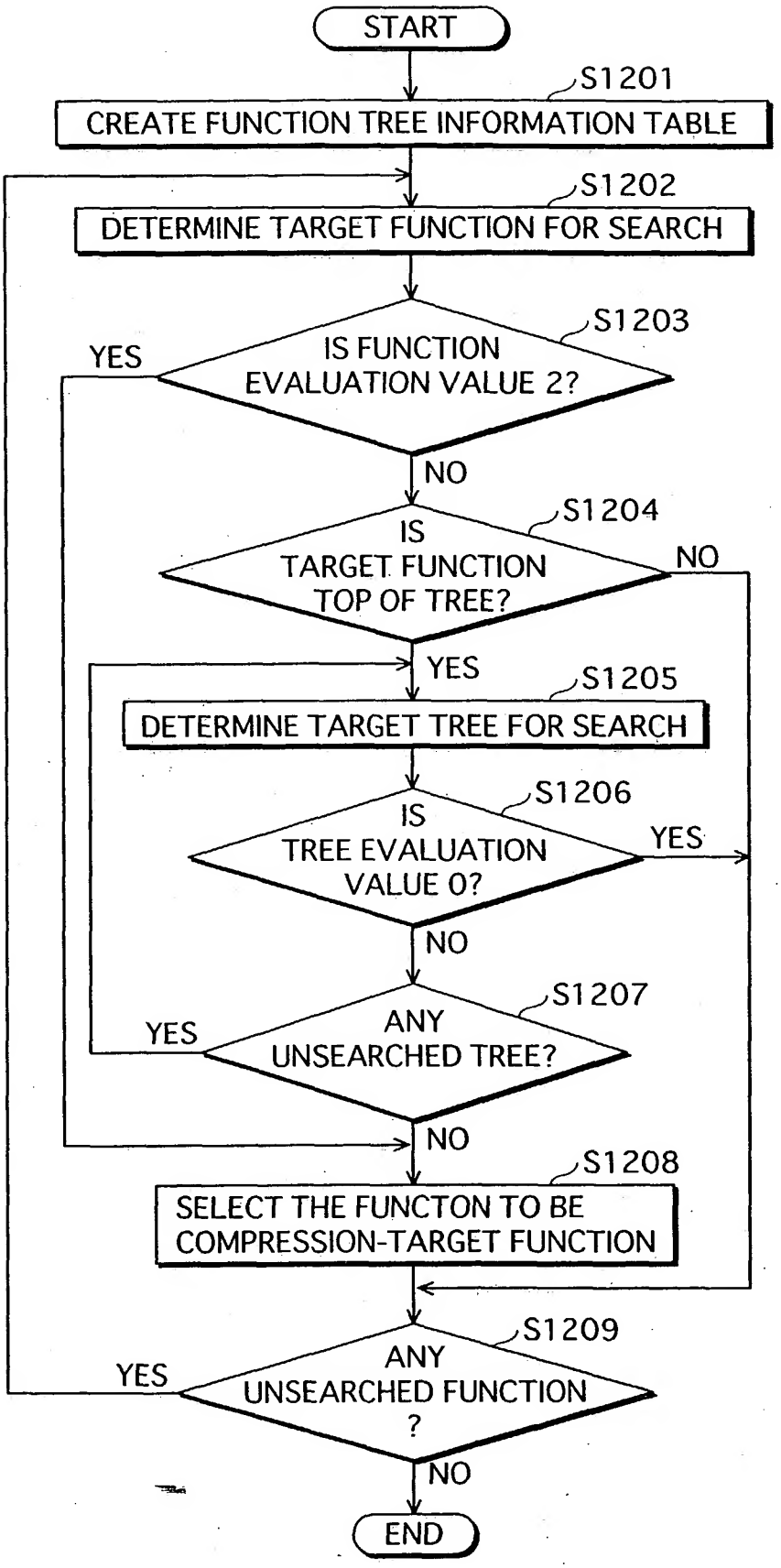

START
S1201
CREATE FUNCTION TREE INFORMATION TABLE
S1202
DETERMINE TARGET FUNCTION FOR SEARCH
S1203
IS FUNCTION EVALUATION VALUE 2?
YES
NO
S1204
IS TARGET FUNCTION TOP OF TREE?
NO
YES
S1205
DETERMINE TARGET TREE FOR SEARCH
S1206
IS TREE EVALUATION VALUE 0?
YES
NO
S1207
ANY UNSEARCHED TREE?
YES
NO
S1208
SELECT THE FUNCTON TO BE COMPRESSION-TARGET FUNCTION
S1209
ANY UNSEARCHED FUNCTION ?
YES
NO
END

FIG.24

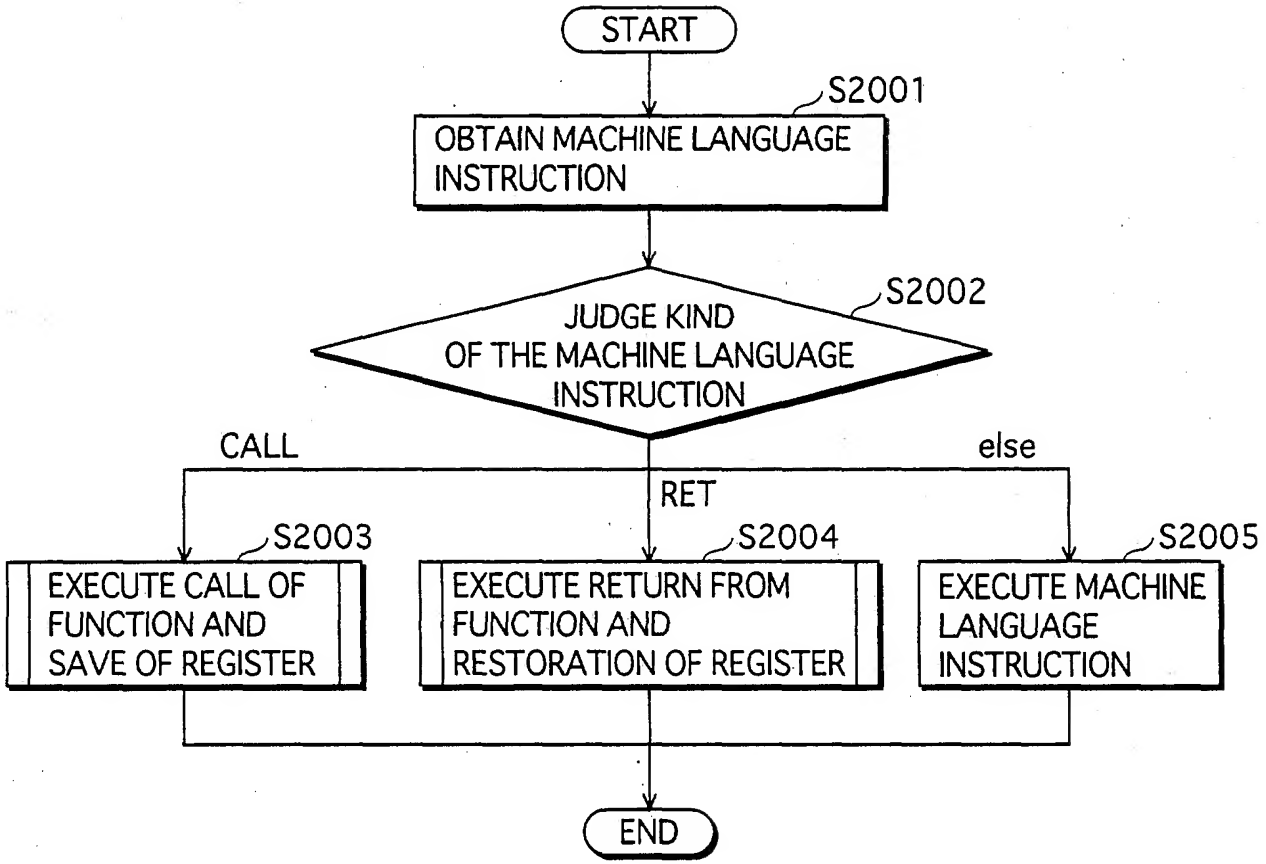

START
S2001
OBTAIN MACHINE LANGUAGE INSTRUCTION
S2002
JUDGE KIND OF THE MACHINE LANGUAGE INSTRUCTION
CALL
RET
else
S2003
EXECUTE CALL OF FUNCTION AND SAVE OF REGISTER
S2004
EXECUTE RETURN FROM FUNCTION AND RESTORATION OF REGISTER
S2005
EXECUTE MACHINE LANGUAGE INSTRUCTION
END

FIG.25

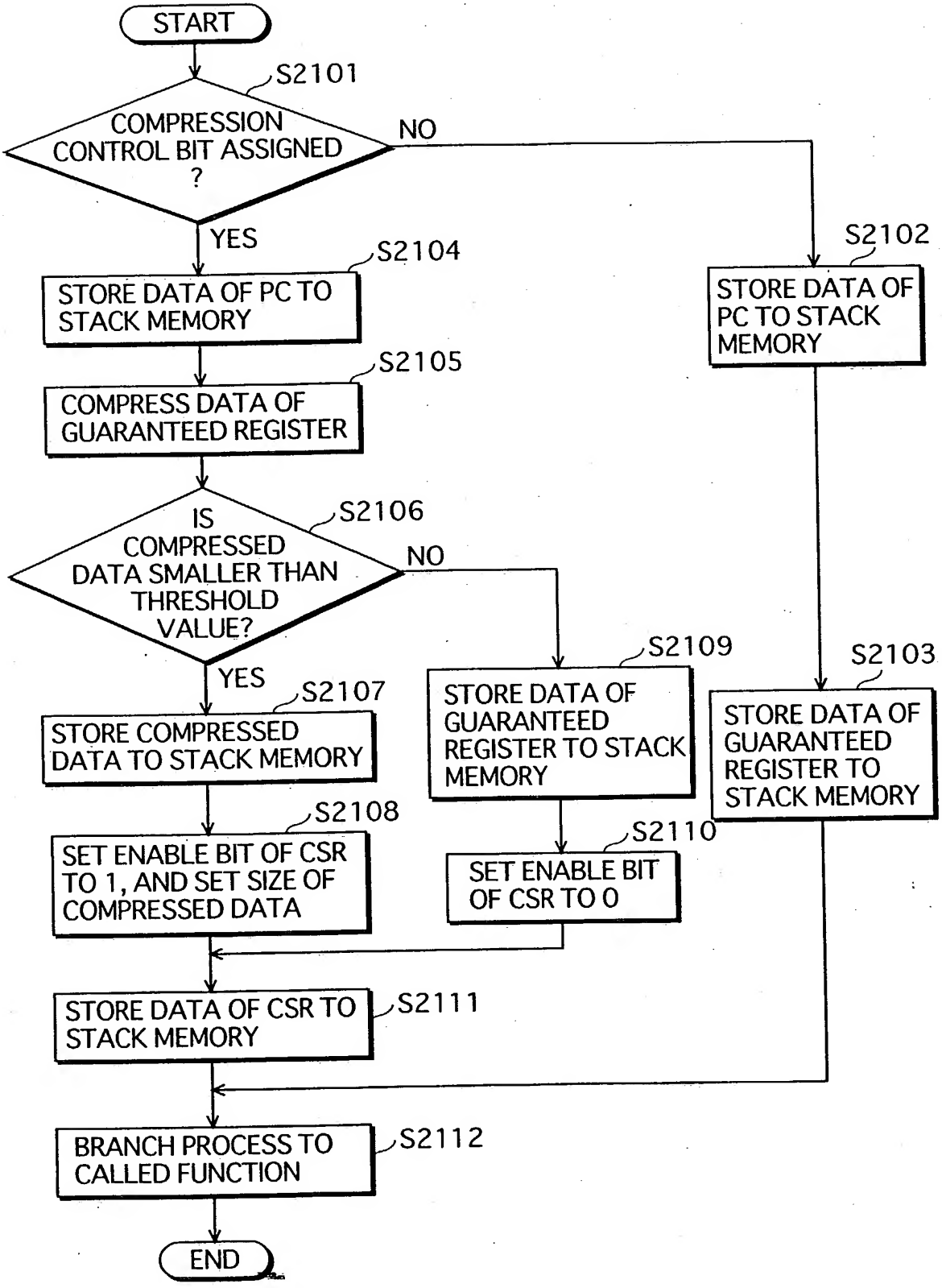
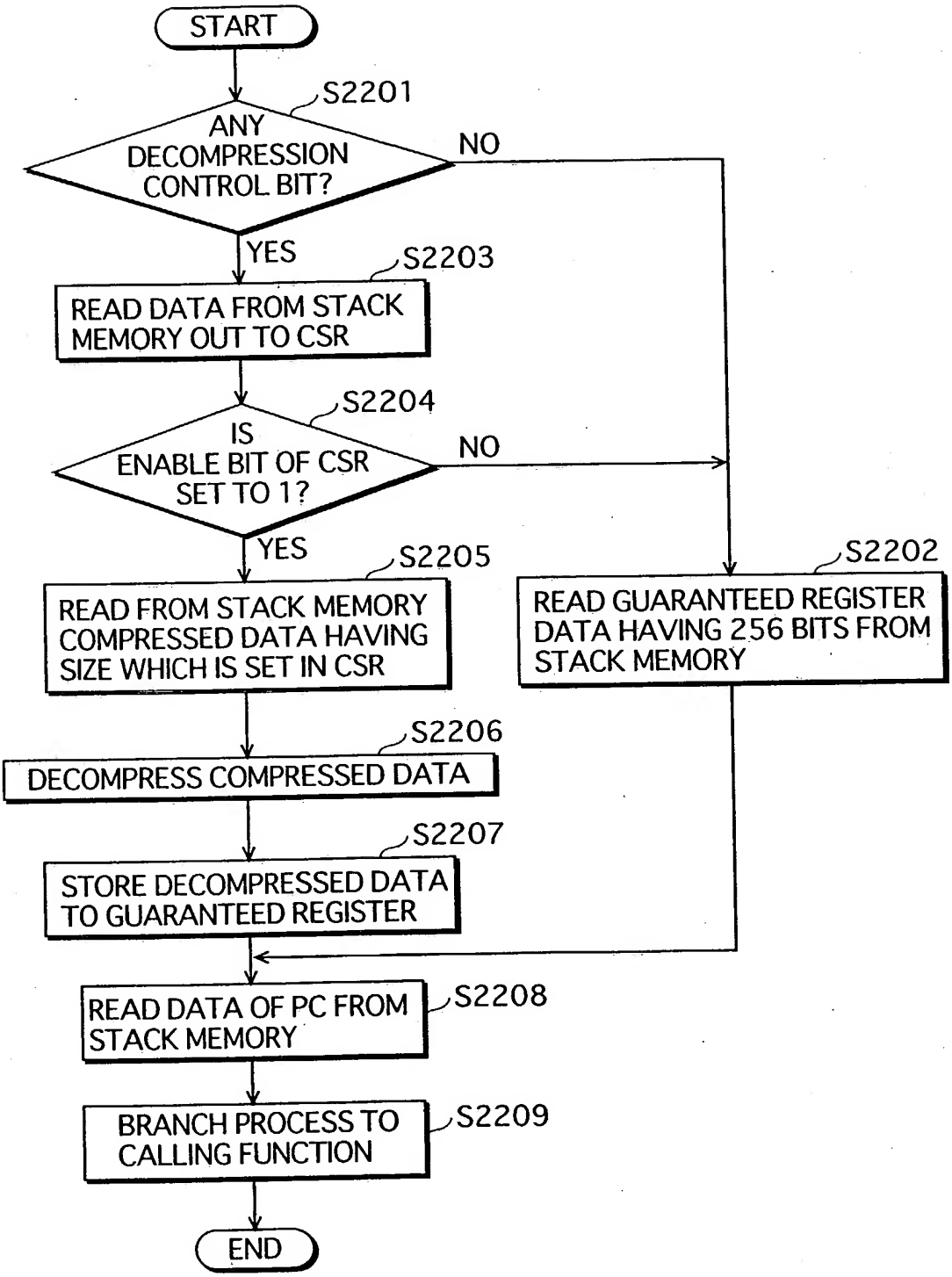

START
S2101
COMPRESSION CONTROL BIT ASSIGNED ?
NO
YES
S2102
STORE DATA OF PC TO STACK MEMORY
S2104
STORE DATA OF PC TO STACK MEMORY
S2105
COMPRESS DATA OF GUARANTEED REGISTER
S2106
IS COMPRESSED DATA SMALLER THAN THRESHOLD VALUE?
NO
YES
S2107
STORE COMPRESSED DATA TO STACK MEMORY
S2108
SET ENABLE BIT OF CSR TO 1, AND SET SIZE OF COMPRESSED DATA
S2109
STORE DATA OF GUARANTEED REGISTER TO STACK MEMORY
S2110
SET ENABLE BIT OF CSR TO 0
S2103
STORE DATA OF GUARANTEED REGISTER TO STACK MEMORY
S2111
STORE DATA OF CSR TO STACK MEMORY
S2112
BRANCH PROCESS TO CALLED FUNCTION
END

FIG.26


START
S2201
ANY DECOMPRESSION CONTROL BIT?
NO
YES
S2203
READ DATA FROM STACK MEMORY OUT TO CSR
S2204
IS ENABLE BIT OF CSR SET TO 1?
NO
YES
S2205
READ FROM STACK MEMORY COMPRESSED DATA HAVING SIZE WHICH IS SET IN CSR
S2202
READ GUARANTEED REGISTER DATA HAVING 256 BITS FROM STACK MEMORY
S2206
DECOMPRESS COMPRESSED DATA
S2207
STORE DECOMPRESSED DATA TO GUARANTEED REGISTER
S2208
READ DATA OF PC FROM STACK MEMORY
S2209
BRANCH PROCESS TO CALLING FUNCTION
END